Escuela Politécnica Superior

Reconocimiento de intrusos en dispositivos móviles

Máster Universitario en Ciberseguridad

Trabajo Fin de Máster

Autor:
Ismael Piñeiro Ramos

Tutor/es:
Juan Antonio Gil Martínez-Abarca

Septiembre 2019

Universitat d'Alacant
Universidad de Alicante

Índice

1 Introducción

La privacidad, según la definición de la RAE [1], es aquello que tiene la cualidad de privado.

Una información privada es aquella que no deseamos que sea divulgada, y desear que una información no sea divulgada no significa que sea difamatoria o perjudicial. Simplemente pertenecer a la vida privada puede ser suficiente para querer salvaguardar la privacidad de la información.

Todas las personas tenemos información que no deseamos que sea divulgada, y sin duda, el dispositivo con más información de carácter privado que hoy en día tenemos las personas es el teléfono móvil. En los teléfonos móviles almacenamos los números de teléfonos de nuestras personas más allegadas, la aplicación de nuestro banco e incluso conversaciones privadas a través de aplicaciones de mensajería.

Los retos que supone la privacidad de la información se afrontan con la seguridad informática, en concreto mediante la protección de la información.

Para proteger la información se utilizan diferentes mecanismos, por ejemplo, algoritmos como AES o el protocolo SSL son ampliamente utilizados para la protección de las comunicaciones. Además de proteger la información aplicando técnicas criptográficas, también se debe autenticar y autorizar a los usuarios y/o sistemas.

Generalmente, para autenticar y autorizar si un usuario tiene acceso a la información, se hace uso de una combinación de nombre de usuario y contraseña. De esta manera, el sistema puede permitir o denegar el acceso.

En los dispositivos móviles la forma más común de garantizar que un usuario tiene acceso a la información es mediante un código PIN o un patrón, siendo esto algo opcional. Al igual que ocurre con la identificación basada en usuario y contraseña, si un tercero consigue acceso al sistema con la sesión iniciada tendría acceso a toda la información del

mismo. Además, debido a la naturaleza del dispositivo móvil, es sencillo que nos sustraigan el teléfono mientras lo estamos utilizando y si el dispositivo no cuenta con la correcta protección el ladrón tendría acceso a toda la información.

El siguiente trabajo investiga la implementación de una capa de seguridad en los teléfonos móviles Android a partir de una aplicación que utiliza diferentes sensores, como la cámara del dispositivo, para realizar una detección de intrusos y aplicar diferentes técnicas de defensa.

2 Estado del arte

En los dispositivos móviles se utilizan principalmente dos sistemas operativos: Android e iOS. Siendo el primero de ellos el más utilizado del mercado, utilizado por más del 75% de la población mundial según StatCounter.com [2].

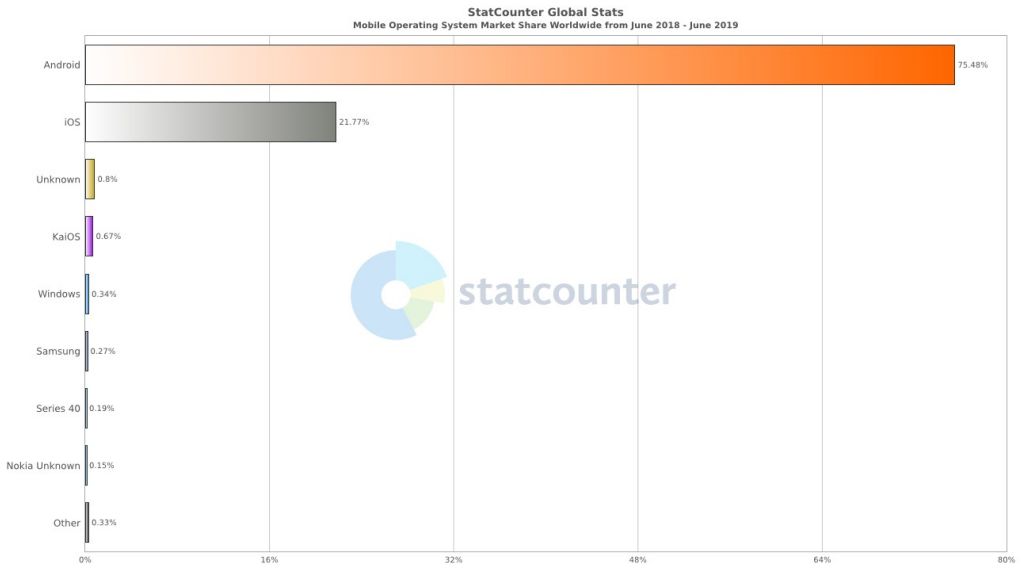

Figura 1: Cuota de mercado de los sistemas operativos móviles en todo el mundo (StatCounter.com)

Por este motivo, todas las pruebas e investigaciones se van a realizar sobre el sistema operativo de Google, aunque serían igualmente aplicables a otro sistemas operativos o dispositivos siempre que cuenten con los sensores necesarios.

Los sistemas operativos Android tienen diferentes mecanismos de seguridad, como por ejemplo, el cifrado de la partición de datos del usuario. Concretamente en la versión 5.0 de Android (Lollipop) se introdujo la característica de cifrado completo del disco, la cuál está siendo reemplazada por el cifrado basado en ficheros introducido en la versión 7.0 (Nougat) [3].

La mayoría de fabricantes de teléfonos Android utilizan la característica introducida en la versión 7.0, suele funcionar en tiempo de arranque descifrando los datos del usuario. Por lo que cuando un teléfono está encendido, la única prevención de un acceso no deseado al sistema operativo frente a un intruso a nivel físico es la pantalla de bloqueo.

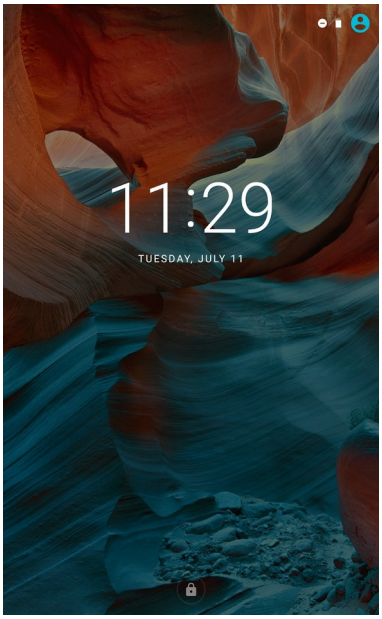

Figura 2: Ejemplo de pantalla de bloqueo de un teléfono Android

En todos los teléfonos Android se incorporan tres principales mecanismos para proteger el dispositivo en la pantalla de bloqueo, los cuales son: la utilización de un PIN, una contraseña o un patrón de desbloqueo.

- **PIN o contraseña**: utilizar un código PIN o una contraseña es el método más común de desbloqueo de un terminal Android, la diferencia entre estos dos métodos es que el código PIN es numérico, mientras que la contraseña es alfanumérica.

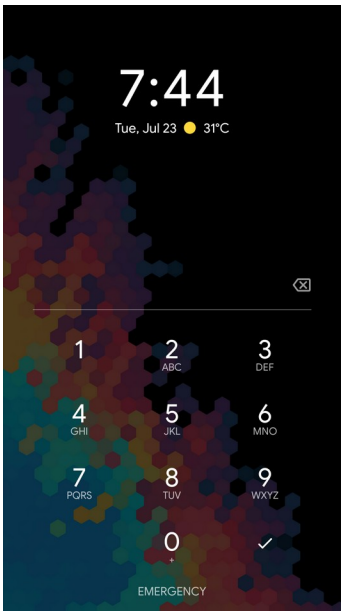

Figura 3: Desbloqueo mediante código PIN

- **Patrón**: es similar a la utilización de un código PIN, pero hace uso de la pantalla táctil para mediante la introducción de gestos táctiles autenticar al usuario.

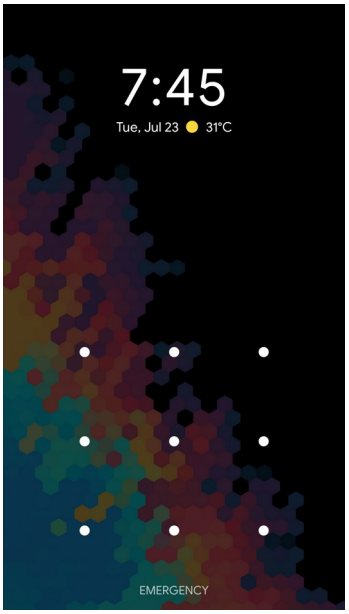

Figura 4: Desbloqueo mediante patrón

Además de estos mecanismos, en los dispositivos más modernos de gama media en adelante se suele incorporar el mecanismo de desbloqueo mediante la huella dactilar, esto se consigue gracias a la incorporación de nuevos sensores.

Todos estos mecanismos protegen un teléfono móvil frente a una persona no autorizada, sin embargo, estos mecanismos no protegen cuando el dispositivo ya se encuentra desbloqueado.

Como un teléfono móvil es algo que siempre se lleva encima y se utiliza en cualquier lugar es sencillo que sea sustraído, por ello se investigan técnicas para prevenir el robo. Por ejemplo, Ericsson ha publicado una nueva patente [4] que intenta evitar el robo de los teléfonos móviles reconociendo al dueño del dispositivo a través de su pulso cardíaco.

Si un dispositivo detecta que un usuario no autorizado intenta cogerlo el dispositivo empezaría a vibrar fuertemente para dificultar su sujeción. A esto lo han llamado fricción adaptativa.

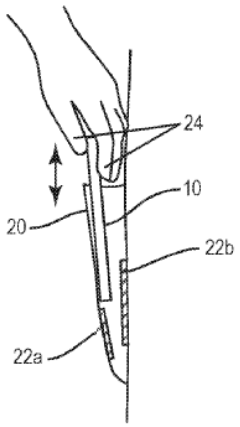

Figura 5: Imagen de la patente de Ericsson

Los sensores, no solo pueden ser utilizados para proteger el teléfono o permitir una serie de funcionalidades en el sistema operativo o en las diferentes aplicaciones, los cibercriminales también pueden aprovecharse de ellos para obtener información.

Investigadores de la universidad de Cornell han realizado un estudio en el que muestran como sería posible obtener las pulsaciones de teclado realizadas por el usuario a partir de las vibraciones producidas, estas vibraciones serían obtenidas a partir del acelerómetro y

del giroscopio. Esta información se procesaría con técnicas de inteligencia artificial y, por ejemplo, se podría obtener el código PIN de un dispositivo cuando el usuario lo teclee. [5]

3 Objetivos

Los objetivos principales de este trabajo son:

- Realizar un estudio de los diferentes sensores disponibles en un dispositivo móvil.
- Reconocer los diferente datos que se pueden obtener de un dispositivo móvil para detectar intrusos.
- Entrenar y aplicar diferentes algoritmos de aprendizaje automático.
- Obtener información sobre la API de Android y los posibles usos a aplicar en un sistema de detección de intrusos.
- Ser capaz de identificar a un intruso con un teléfono móvil Android mediante en el uso de los sensores.
- Implementar diferentes mecanismos de seguridad una vez se ha detectado un intruso.

4 Metodología y herramientas de apoyo

Para la realización del proyecto se ha utilizado una metodología de desarrollo ágil, dividida en cuatro bloques diferentes: investigación, desarrollo, documentación y refinamiento.

Durante la fase de investigación se ha estudiado la funcionalidad y las diferentes posibilidades de los sensores y aquellas herramientas de apoyo útiles para conseguir los objetivos planteados.

4.1 Metodología ágil

Para la planificación utilizando metodología ágil se ha dividido el tiempo en sprints de 1 semana cada uno, estimando una media de 12 horas de carga de trabajo para cada sprint.

- Sprint 1 (Investigación): Perfilar la idea principal y los objetivos a plantear.
- Sprint 2 (Investigación): Explorar las posibilidades de la plataforma Android y su API.
- Sprint 3 (Investigación): Profundizar sobre los sensores y otras herramientas de apoyo como la cámara del dispositivo.
- Sprint 4 (Investigación): Identificar sobre técnicas de detección y de notificación.
- Sprint 5 (Desarrollo): Creación de proyecto y preparación de herramientas necesarias.
- Sprint 6 (Desarrollo): Implementación y pruebas del uso de sensores.
- Sprint 7 (Desarrollo): Uso de técnica de reconocimiento de intrusos.
- Sprint 8 (Desarrollo): Creación de interfaz de usuario.
- Sprint 9 (Desarrollo): Implementación de técnicas de notificación .
- Sprint 10 (Desarrollo): Mejora de la detección de intrusos mediante uso de sensores.

- Sprint 11 (Documentación): Introducción, estado del arte y objetivos de la memoria.

- Sprint 12 (Documentación): Análisis de los sensores en Android

- Sprint 13 (Documentación): Técnicas de detección de intrusos y de defensa.

- Sprint 14 (Documentación): Conclusiones

- Sprint 15 y 16 (Refinamiento): Revisión del documento, pruebas de la aplicación y arreglo de fallos.

Hay que tener en cuenta que en muchos casos existe paralelismo en la planificación, por ejemplo la investigación es constante, aunque el mayor esfuerzo se realiza según la planificación.

4.2 Análisis de sensores en Android

La interacción más común de un usuario con un dispositivo Android es a partir de la pantalla táctil. Sin embargo, esta no es la única manera en la que un dispositivo puede recibir información del exterior, ya que se dispone de una serie de sensores que pueden utilizarse para este fin.

En la mayoría de los teléfonos móviles Android se incorporan diferentes sensores que miden el movimiento, la orientación y diversas condiciones ambientales. Además, los sensores pueden clasificarse según como son implementados. Un sensor puede ser una pieza física de hardware o puede valerse de otras piezas para derivar una medición mediante software.

La API de Android [6] expone facilidades de uso para los siguientes principales sensores desde la versión de Android 4.0 o nivel de API 14:

Tipo de sensor	Sensor	Hardware	Software
Sensores de movimiento	Acelerómetro	X	
	Gravedad	X	X
	Aceleración lineal	X	X
	Vector de rotación	X	X
Sensores de posición	Giroscopio	X	
	Magnetómetro	X	
	Orientación		X
	Presión	X	
	Proximidad	X	
	Vector de rotación	X	X
Sensores de entorno	Temperatura ambiental	X	
	Luz	X	
	Humedad relativa	X	

En la tabla anterior se pueden ver los diferentes tipos de sensores soportados en la plataforma Android y como pueden ser implementados.

Además de los sensores anteriores, también existen otros como el GPS, el sensor de huellas dactilares, el micrófono y la cámara del dispositivo, pero estos no son considerados como parte de los sensores en la API de Android [7].

A continuación se detalla el funcionamiento de algunos de los sensores más utilizados en Android, incluyendo la Cámara y el GPS.

4.2.1 Acelerómetro

El acelerómetro o sensor de aceleración se utiliza para medir la aceleración que es aplicada sobre el dispositivo, en la medición del acelerómetro se incluye la fuerza de la gravedad.

Matemáticamente, la aceleración detectada por el sensor de aceleración es expresada mediante la siguiente fórmula

$$A_d = -g - (\frac{1}{masa}) \sum F_s$$

La aceleración del dispositivo (A_d) es expresada mediante un vector tridimensional con las fuerzas de aceleración detectadas (incluyendo la gravedad) en los diferentes ejes.

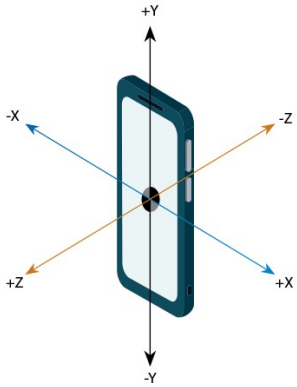

Figura 6: Representación gráfica de un acelerómetro

Como la gravedad se incluye en las medidas obtenidas por el sensor, para obtener una correcta lectura se debe eliminar la gravedad aplicando un filtrado sobre las lecturas del sensor o se puede utilizar el sensor de aceleración lineal.

4.2.2 Sensor de aceleración lineal

El sensor de aceleración lineal funciona igual que el acelerómetro, pero excluyendo la gravedad. De manera conceptual, la aceleración lineal se obtiene a partir de la aceleración real menos la aceleración producida por la gravedad.

Al igual que las medidas obtenidas por el acelerómetro, se obtiene un vector tridimensional con la aceleración en los diferentes ejes, pero excluyendo la gravedad.

Las medidas del sensor de aceleración tienen un margen de error, este margen de error puede ser calibrado situando el dispositivo sobre una zona plana y obteniendo las medidas del desplazamiento, para posteriormente ser restado de las medidas obtenidas.

4.2.3 Giroscopio

El giroscopio es un sensor que sirve para medir la orientación y la velocidad angular. Al igual que los sensores de aceleración la salida producida por el giroscopio es un vector tridimensional, donde la rotación es positiva en el sentido opuesto a las agujas del reloj.

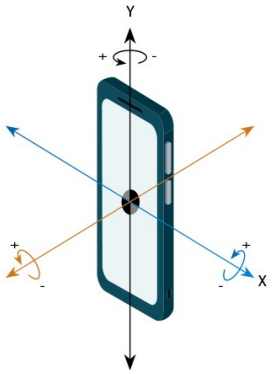

Figura 7: Representación gráfica del giroscopio

La utilización del acelerómetro junto al giroscopio ayudan a detectar con bastante precisión los movimientos que se están aplicando sobre un dispositivo móvil, por ello son ampliamente usados en videojuegos . Por ejemplo, los juegos de carreras utilizan a menudo estos sensores para realizar el control del vehículo a partir de la rotación y la inclinación.

Su uso no es exclusivo de los dispositivos móviles y son ampliamente usados en otros campos, por ejemplo son un elemento fundamental de la aviación o en la tecnología de la realidad virtual.

4.2.4 Sensor de gravedad

El sensor de gravedad es utilizado para medir la aceleración producida por la fuerza de la gravedad, ayudando a determinar la orientación relativa de un dispositivo en el espacio. Generalmente es un sensor implementado por software y que se apoya en otros sensores para realizar la medición.

En las primeras generaciones de dispositivos móviles Android se utilizaban las medidas obtenidas por el acelerómetro aplicando una minimización sobre la aceleración lineal, ya que estos valores tienden a cero. Esta técnica era funcional, sin embargo, con la incorporación de los giroscopios en los dispositivos móviles más modernos, se implementa un sensor virtual que otorga unas medidas mucho más precisas de la gravedad debido a la posibilidad de obtención de la inclinación [8].

Debido a su naturaleza, si un dispositivo se encuentra sobre una mesa completamente en reposo, la salida del sensor de gravedad debe ser igual que la salida producida por el sensor de aceleración.

4.2.5 Sensor de vector de rotación

El sensor de vector de rotación determina la orientación respecto a un ángulo para cada uno de los ejes en un marco tridimensional, donde el dispositivo ha girado a través de un ángulo, definido como θ, en alguno de los ejes. La rotación para cada uno de los ejes es expresada mediante las siguientes fórmulas:

$$R_x = x * \sin(\frac{\theta}{2}) \qquad R_y = y * \sin(\frac{\theta}{2}) \qquad R_z = z * \sin(\frac{\theta}{2})$$

Este sensor se implementa a través de software y utiliza la información de otros sensores físicos como el acelerómetro y el giroscopio. Aunque su forma matemática puede resultar compleja es la forma más eficiente y precisa de determinar la orientación de un dispositivo.

4.2.6 Sensores de entorno

Los sensores de entorno permiten obtener información acerca de las condiciones ambientales cerca de un dispositivo. En Android se proporciona una interfaz para trabajar con cuatro sensores, que se utilizan para obtener información de la humedad del ambiente, la iluminación, la presión ambiental y la temperatura.

Todos estos sensores son implementados mediante hardware y no se encuentran incorporados en todos los dispositivos Android. Generalmente el sensor de ambiente que suele estar presente es el sensor de iluminación, ya que es el utilizado para controlar la intensidad del brillo de la pantalla según la luz del entorno.

4.2.7 GPS

El GPS (Global Positioning System) es un sistema de navegación que permite obtener información acerca de la geolocalización de un dispositivo en un momento determinado utilizando diferentes frecuencias de radio y un receptor.

Este sistema esta controlado por la fuerza aérea de los Estados Unidos y aunque existen otros sistemas como el Galileo, perteneciente a la Unión Europea, el sistema GPS es el más utilizado a nivel global.

Además de la utilización del GPS, se suelen utilizar las redes Wi-Fi o las celdas de la red móvil para identificar la geolocalización de un dispositivo Android.

Si se quiere utilizar la API de alto nivel de localización se deben activar y configurar los servicios de Google Play en la aplicación [9].

4.2.8 Cámara

La API de Android incorpora las funcionalidades necesarias para el control de la cámara de un dispositivo, de manera que permite realizar imágenes y vídeos. En la plataforma Android existen tres API diferentes para el control de la cámara:

- Camera API: Se trata de la versión antigua y esta marcada como desfasada a partir de la versión 5.0 de Android (nivel de API 21).

- Camera2 API: Nueva versión recomendada para las nuevas aplicaciones, es compatible con los dispositivos que cuenten con versiones de Android 5.0 o superiores.

- CameraX: Biblioteca de compatibilidad para facilitar el desarrollo, haciendo que sea más fácil de utilizar la cámara y no tener que preocuparse de problemas de compatibilidad entre diferentes dispositivo.

Para poder utilizar la cámara de un dispositivo existen una serie de consideraciones que se deberían tener en cuenta:

- Debemos asegurarnos de que el dispositivo tiene una cámara y esta puede ser utilizada.

- Para poder almacenar las imágenes en la memoria interna del teléfono debemos tener los permisos necesarios.

- A partir de Android 9 (nivel de API 28) las aplicaciones y/o servicios solo pueden acceder a la cámara si están funcionando en primer plano.

5 Cuerpo del trabajo

5.1 Técnicas de detección de intrusos

Los sensores pueden ser utilizados para detectar el movimiento que realiza el usuario, saber la posición en la que se encuentra el dispositivo o incluso obtener información sobre el entorno en el que se encuentran. Usando la información sobre estos sensores se pueden aplicar medidas de seguridad sobre un dispositivo.

Para realizar la detección de un intruso o acceso no deseado a un terminal Android, se exploran dos diferentes posibilidades: el reconocimiento del movimiento y el reconocimiento facial.

5.1.1 Reconocimiento del movimiento

La detección de un intruso mediante el reconocimiento del movimiento se puede realizar de una manera básica mediante el sensor de aceleración. En un escenario en el que el usuario se encuentra utilizando el teléfono, bien hablando con el o simplemente jugando, si se detecta un movimiento brusco se puede significar que el teléfono ha sido sustraído.

Para detectar un movimiento brusco utilizando el sensor de aceleración se calcula la aceleración mediante la raíz cuadrada de la suma de los cuadrados del vector tridimensional devuelto por el sensor.

$$A = \sqrt{(V_0^2 + V_1^2 + V_2^2)}$$

Después se compara la aceleración lineal obtenida con un umbral establecido, en las pruebas realizadas el mejor umbral encontrado esta entre los valores de 8 y 12, aunque para un funcionamiento óptimo se debería ajustar por dispositivo.

Cuando se ha detectado un movimiento brusco, la primera medida de seguridad a realizar sería bloquear el terminal, de manera que sin el código o pin de acceso no sea posible acceder a la información que contiene.

Una forma más avanzada de realizar la detección de intrusos sería utilizar técnicas de inteligencia artificial y aprendizaje automático. Por ejemplo, entrenar una red neuronal con la información obtenida del acelerómetro y del giroscopio, con el fin de conseguir perfilar el usuario y sus movimientos más comunes.

Las personas seguimos una serie de patrones involuntarios a la hora de manejar un dispositivo móvil, esto es principalmente debido a la memoria muscular. Por ello, si se

perfilan los movimientos habituales de un usuario, se podrá detectar a través de la red neuronal entrenada si se trata del usuario habitual o si se trata de un posible intruso.

Por último, la plataforma de Android aporta una API de alto nivel que utiliza los sensores disponibles y puede ser utilizada para el reconocimiento del movimiento de una manera inteligente. Esta API es conocida como la API de reconocimiento de actividades [10].

Sin embargo, esta API se principalmente utilizada para detectar las siguientes actividades físicas derivadas del movimiento:

- El dispositivo se encuentra en un vehículo.
- El dispositivo se encuentra en una bicicleta.
- El dispositivo lo lleva una persona andando o corriendo.

Un caso práctico utilizando esta API para aplicar seguridad ante el robo de un dispositivo móvil sería el detectar un cambio de movimiento de ir andando o estar parado y de repente, estar corriendo. Este cambio de actividad puede indicar que el teléfono ha sido sustraído, aunque se debe permitir una configuración, de manera que el usuario pueda establecer horarios o el momento en que el sale a correr.

5.1.2 Reconocimiento facial

Una técnica más avanzada que el reconocimiento de movimiento para la detección de intrusos es el reconocimiento facial. Tomar una fotografía cuando el teléfono esta siendo utilizado y reconocer si se trata de un usuario habitual o un intruso es eficaz, pero puede suponer un problema en cuanto a rendimiento se refiere.

El primer paso para poder identificar a un intruso, es poder identificar una cara en una fotografía, para ello se utiliza el algoritmo HOG (*Histogram of Oriented Gradients*) publicado en el año 2005 [11] para generar un conjunto de entrenamiento.

El funcionamiento del algoritmo HOG de manera resumida es el siguiente [12]:

1. Convertir la imagen a escala de grises

2. Para cada pixel en la imagen se deben analizar sus pixeles cercanos y determinar en que dirección el color se va oscureciendo.

3. Obtener los gradientes, para conseguir los gradientes hay que reemplazar los pixeles por una flecha en la dirección determinada en el paso 2.

4. Divide la imagen en cuadrados pequeños (generalmente de 16x16) y de cada cuadrado contar los gradientes para cada dirección. Reemplazar el cuadrado en la imagen por los gradientes que tengan el mayor peso, es decir, de los que tengan más flechas apuntando a una dirección concreta.

Figura 8: Imagen original de entrada

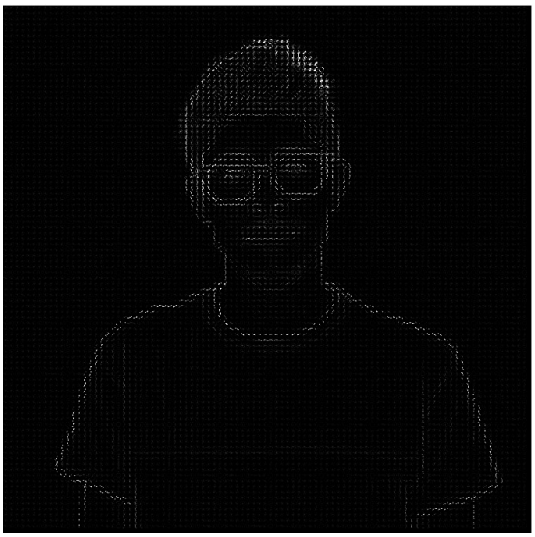

Figura 9: Imagen procesada con HOG

Como se puede ver en la imagen procesada con el algoritmo HOG se obtiene los contornos de lo que aparece en la imagen, esto es gracias a los gradientes que indican el cambio de luz.

El algoritmo HOG se aplica a un conjunto de entrenamiento y se extrae un patrón de la zona ha detectar.

Figura 10: Ejemplo de patrón HOG facial

Para realizar la detección se debe comparar el patrón generado con una imagen de entrada, el patrón ayudará a delimitar la zona de la coincidencia si la hubiese.

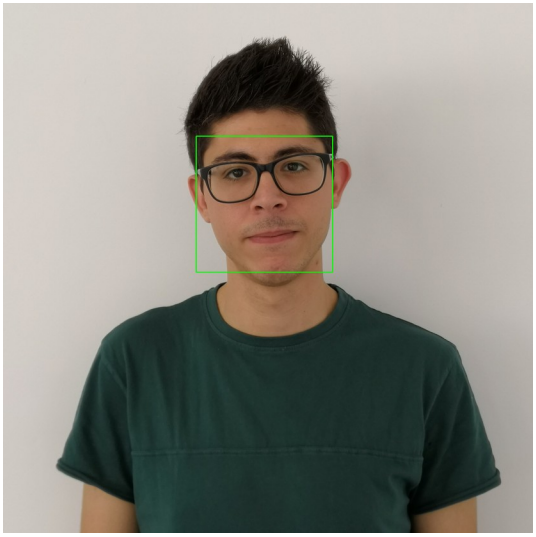

Figura 11: Reconocimiento de rostro delimitado con patrón HOG

Con la detección facial no es suficiente para identificar un intruso, además de detectar caras en imágenes se debe ser capaz de comparar dos rostros diferentes. Para la comparación se usan unas características o *landmarks* que pueden ser obtenidas de diferentes maneras.

Generalmente se establecen una serie de puntos que existen en todos los rostros y se entrena un algoritmo que permite encontrar estos puntos en una imagen [13].

Figura 12: Ejemplo de características extraídas

Estas características extraídas sirven para ajustar la posición de la imagen y poder realizar la comparación entre ellas.

La manera más sencilla de realizar la comparación entre dos conjuntos de características diferentes sería el calcular las similitudes, por ejemplo: medir el tamaño de las orejas, medir el tamaño de la boca, medir la distancia entre los ojos, etc. También se puede realizar la distancia entre cada característica, por ejemplo aplicar la distancia euclidiana.

Sin embargo, el método más eficaz para la comparación de características es utilizar algoritmos de *deep learning,* como por ejemplo, entrenar una red Neuronal Convolucional Profunda (o CNN*,* de *Convolutional Neural Network*) que aprenda a tomar diferentes medidas que se compararan posteriormente.

El uso de algoritmos de redes neuronales es algo bastante complejo para ser tratado en este trabajo, por ello no se va a detallar en profundidad su funcionamiento.

En la plataforma Android existe una librería de alto nivel que facilita el uso de *machine learning* para resolver unas tareas concretas. Para utilizar esta librería no es necesario un conocimiento previo sobre redes neuronales, con solo unas líneas de código se pueden realizar tareas de reconocimiento facial o reconocimiento de texto.

La librería en cuestión es ML Kit [14], se encuentra incluida dentro del paquete de Firebase y actualmente esta en fase beta. Puede funcionar en dos modos diferentes: en el dispositivo o en la nube. Dependiendo del modo, se incluye una funcionalidad u otra.

La funcionalidad de detección facial esta disponible tanto en al nube como en el propio dispositivo y obtiene un total de 133 puntos característicos o *landmarks* diferentes. Sin embargo, no realiza el reconocimiento o comparación de rostros, únicamente realiza la detección facial.

Otra librería que puede ser utilizada en dispositivos Android es TensorFlow Lite [15], la hermana pequeña de la librería de *deep learning* para dispositivos móviles y embebidos de Google. Además TensorFlow Lite es compatible con ML Kit, ya que los modelos generados y entrenados se pueden utilizar en ML Kit.

5.2 Técnicas de defensa

Una vez se ha detectado un intruso, se debe realizar una acción en consecuencia que sirva como técnica de defensa. Para aportar seguridad estos mecanismos deben proteger la información y poder notificar de los diferentes eventos que ocurran.

5.2.1 Bloqueo automático

Como se ha mencionado anteriormente, la única seguridad incorporada en los dispositivos Android de manera nativa frente a un intruso sería la pantalla de bloqueo. Por ello, la primera técnica de defensa que se debe implementar es, que si un intruso consigue el teléfono desbloqueado el sistema debe de ser capaz de realizar un bloqueo automático del dispositivo.

En Android para poder realizar esta característica se necesitan unos permisos especiales de administración del dispositivo [16]. Este tipo de permiso generalmente es utilizado en aplicaciones empresariales internas, para poder administrar los dispositivos a nivel de

sistema a través de una aplicación.

Entre los usos más comunes de los permisos de administración del dispositivo se encuentran:

- Aplicar políticas de contraseñas (fuerza de contraseñas, intentos fallidos, caducidad…)

- Realizar un formateo del dispositivo restableciendo los valores de fábrica del dispositivo.

- Bloquear el dispositivo

Cuando una aplicación va a utilizar esta característica, es necesario solicitar explícitamente estos permisos al usuarios, como se puede ver en la siguiente imagen.

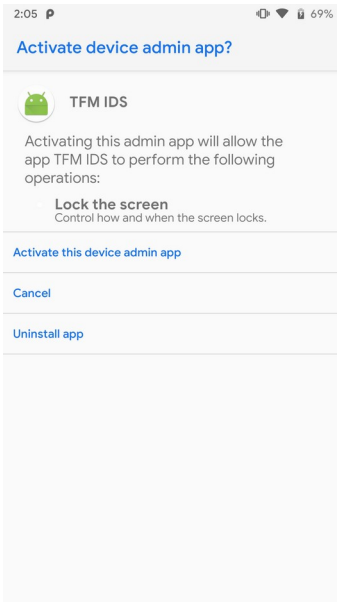

Figura 13: Pantalla de solicitud de permisos de administración

Una vez se han solicitado los permisos se tendría la capacidad para realizar el bloqueo del dispositivo de manera programática como consecuencia de un evento.

5.2.2 Sistema de notificaciones

La capacidad de emitir notificaciones es una técnica de defensa utilizada en todos los sistemas de detección de intrusos.

Existen múltiples vías por las que se pueden lanzar notificaciones, lo más habitual sería enviar un mensaje de texto (SMS) o un correo electrónico, aunque también se podrían realizar llamadas o utilizar sistemas de mensajería como Slack o Telegram.

El hecho de utilizar notificaciones vía correo electrónico u otros sistemas que requieren de una conexión a Internet puede suponer un problema, ya que puede que no se disponga de conexión en todo momento. Por eso el enviar mensajes de texto es uno de los métodos más eficaces, con sus limitaciones.

Dependiendo del canal de comunicación, el envío de la información puede verse limitado, por ejemplo con los SMS no se podrían enviar archivos adjuntos, aunque si con los MMS o en un correo electrónico.

Una información relevante que debería enviarse en toda notificación sería la localización del dispositivo, de esta manera se podría realizar un seguimiento de donde se encuentra y así poder recuperarlo.

5.2.3 Alerta sonora

Otra técnica de defensa implementada es una alerta en forma de sonido, de manera que cuando se detecta un intruso se emite un audio avisando de que el teléfono ha sido robado.

5.3 Desarrollo e implementación

A continuación se muestran algunos fragmentos de código utilizados en el desarrollo de la aplicación, además se explica su funcionalidad.

5.3.1 Detección de movimiento brusco

Una de las maneras de obtener información de los sensores es suscribirse a los eventos de cada lectura del sensor. Para ello es necesario obtener una instancia del SensorManager, a continuación se muestra el código necesario:

```
sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
sensor = sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION);
sensorManager.registerListener(sensorListener, sensor,
SensorManager.SENSOR_DELAY_NORMAL);
```

El código anterior muestra como registrar un escuchador en el sensor de aceleración lineal. A partir de este escuchador sería posible realizar detecciones de movimiento, como por ejemplo detectar un movimiento brusco

```
public void onSensorChanged(SensorEvent sensorEvent) {
     if (sensorEvent.sensor.getType() == Sensor.TYPE_LINEAR_ACCELERATION) {
            float x = sensorEvent.values[0];
            float y = sensorEvent.values[1];
            float z = sensorEvent.values[2];

            float acceleration = (float) Math.sqrt(x * x + y * y + z * z);
            if (acceleration > MOTION_THRESHOLD) {
                App.debug("Abrupt motion detected, acceleration: "+acceleration);
                // do things like send notifications, block phone
            }
     }
}
```

5.3.2 Cámara en segundo plano

Para poder realizar fotografías con un dispositivo Android en cualquier momento y sin que el usuario se de cuenta se ha implementado una clase que permite realizar fotografías en segundo plano a partir de la API Camera2.

Las fotografías se realizan con la cámara delantera, para ello se usa el siguiente fragmento de código para obtener el identificador de la cámara del dispositivo, en caso de que el dispositivo carezca de cámara delantera se lanzará una excepción.

```
private String getFrontCameraId() throws CameraAccessException {
```

```
        String[] idList = cameraManager.getCameraIdList();

        for(String id : idList) {
                cameraCharacteristics = cameraManager.getCameraCharacteristics(id);
                Integer lensFacing =
    cameraCharacteristics.get(CameraCharacteristics.LENS_FACING);

                if(null != lensFacing && lensFacing ==
    CameraCharacteristics.LENS_FACING_FRONT) {
                    return id;
                }
        }

        throw new CameraAccessException(CameraAccessException.CAMERA_ERROR);
    }
```

A partir del id obtenido se abre la cámara con un callback que permitirá realizar las fotografías.

```
cameraManager.openCamera(frontCameraId, cameraStateCallBack, null);
```

Para obtener una fotografía nítida y de buena calidad es necesario realizar una calibración de la cámara. Al igual que cuando se abre la aplicación de la cámara de cualquier dispositivo Android, tarda unos segundos en enfocar, al utilizarla en segundo plano ocurre lo mismo.

Cuando la cámara se encuentra configurada se almacena el tiempo actual, y hasta que transcurra el tiempo establecido para la calibración, se van tomando imágenes que son desechadas hasta calibrar la cámara.

```
private CameraCaptureSession.StateCallback sessionStateCallback = new
CameraCaptureSession.StateCallback() {
     ...
     @Override
     public void onConfigured(@NonNull CameraCaptureSession cameraCaptureSession) {
            ...
            captureStartTime = System.currentTimeMillis();
            ...
     }
     ...
};

...

imageReader = ImageReader.newInstance(width, height, ImageFormat.JPEG, 24);
imageReader.setOnImageAvailableListener(new ImageReader.OnImageAvailableListener() {
     @Override
     public void onImageAvailable(ImageReader imageReader) {
     Image image = imageReader.acquireNextImage();
     if(image == null) {
```

```
            return;
        }

        if(System.currentTimeMillis() > (captureStartTime + CALIBRATION_DELAY)) {
            App.debug("Photo taken!");
            ...
            image.close();
            imageReader.close();
            cameraDevice.close();
            return;
        }
        ...
    }, null);
```

5.3.3 Acciones del sistema

El sistema Android tiene la capacidad de informar a las aplicaciones instaladas sobre acciones ocurridas en el sistema,. Algunas de estas acciones hacen referencia a la interacción del usuario con el dispositivo y otras simplemente informan sobre cambios en el estado del teléfono. Por ejemplo, para ser notificados cuando el dispositivo se encienda debemos extender de la clase BroadcastReceiver

```
public class BootCompleteReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if(action != null && action.equals(Intent.ACTION_BOOT_COMPLETED)) {
            // Do things… like start security service
            startSecurityService(context);
        }
    }

}
```

Será necesario declarar este receptor de la acción en el archivo de manifiesto AndroidManifest.xml para que sea ejecutado

```
<receiver android:name=".receiver.BootCompleteReceiver" android:enabled="true">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>
```

Como es posible que el sistema Android detenga los servicios en segundo plano cuando necesite liberar recursos, una técnica utilizada para asegurar que el servicio de seguridad

esta en funcionamiento es utilizar la acción de usuario presente, como se muestra en el siguiente fragmentos

```
public class UserPresentBroadcastReceiver extends BroadcastReceiver {
     @Override
     public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            if(action != null && action.equals(Intent.ACTION_USER_PRESENT)) {
            if (!ServiceUtils.isServiceRunning(context, SecurityService.class)) {
                String message = "Starting service from UserPresentBroadcastReceiver";
                Toast.makeText(context, message, Toast.LENGTH_LONG).show();
                Intent startServiceIntent = new Intent(context,
SecurityService.class);
                context.startService(startServiceIntent);
            }
        }
    }
}
```

5.3.4 Proveedor de localización

La localización de un dispositivo se suele obtener a mediante la red móvil o el sensor GPS. Para obtener unos resultados óptimos el siguiente fragmento de código muestra como obtener la localización más precisa.

```
public Location getBetterAccuracyLastKnown() {
     Location networkLocation = getNetworkLastKnown();
     Location gpsLocation = getGpsLastKnown();

     if(networkLocation == null && gpsLocation == null) {
            return null;
     }

     if(networkLocation == null) {
            return gpsLocation;
     }

     if(gpsLocation == null) {
            return networkLocation;
     }

     if(gpsLocation.getAccuracy() > networkLocation.getAccuracy()) {
            return gpsLocation;
     }

     return networkLocation;

}

public Location getNetworkLastKnown() {
     return locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
}
```

```
public Location getGpsLastKnown() {
     return locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
}
```

5.3.5 Servicio de seguridad

El servicio de seguridad es el encargado de realizar la detección de intrusos con las diferentes técnicas implementadas y aplicar las medidas de defensa ante una detección.

La técnica de defensa de detección principal implementada es el reconocimiento facial y se necesita un mecanismo para realizar las fotografías en segundo plano sin intervención del usuario. Para ello se ha implementado un JobService que se puede planificar para ejecutarse cada determinado tiempo y además evitar ser destruido por el sistema cuando este necesite liberar recursos.

```
public class CameraDetectorJobService extends JobService {

    private static final int MIN_JOB_DELAY = 5 * 1000; // ms
    private static final int MAX_JOB_DELAY = 60 * 1000; // ms

    private AlertsManager alertsManager;

    public static void schedule(Context context, JobScheduler jobScheduler) {
        alertsManager = new AlertsManager(context);
        ComponentName serviceComponent = new ComponentName(context,
CameraDetectorJobService.class);
        JobInfo.Builder builder = new JobInfo.Builder(0, serviceComponent);
        builder.setRequiresDeviceIdle(false);
        builder.setMinimumLatency(MIN_JOB_DELAY);
        builder.setOverrideDeadline(MAX_JOB_DELAY);
        jobScheduler.schedule(builder.build());
    }

    @Override
    public boolean onStartJob(JobParameters jobParameters) {
        if(isPhoneUnlocked()) {
            performCameraDetection(jobParameters);
        }

        return true;
    }

    @Override
    public boolean onStopJob(JobParameters jobParameters) {
        return true;
    }

}
```

El código anterior muestra la implementación básica del JobService y este se planifica a través del servicio de seguridad como se muestra a continuación

```
JobScheduler jobScheduler = getSystemService(JobScheduler.class);
CameraDetectorJobService.schedule(getApplicationContext(), jobScheduler);
```

Además, para evitar que el servicio de seguridad sea destruido se crea una notificación junto al servicio, de manera que aunque se cierre la aplicación el servicio de seguridad podrá seguir funcionando.

```
public class SecurityService extends Service {

    AbruptMotionDetector abruptMotionDetector;

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        super.onStartCommand(intent, flags, startId);
        new StartPythonAsyncTask(getApplicationContext()).execute();
        App.debug("Security Service started");
        startForeground(1000, createNotification());
        startCameraDetectorJob();
        startAbruptMotionDetector();
        return START_STICKY;
    }

    private void startCameraDetectorJob() {
        JobScheduler jobScheduler = getSystemService(JobScheduler.class);
        CameraDetectorJobService.schedule(getApplicationContext(), jobScheduler);
    }

    private void startAbruptMotionDetector() {
        SensorManager sensorManager = (SensorManager)
getApplicationContext().getSystemService(Context.SENSOR_SERVICE);
        abruptMotionDetector = new AbruptMotionDetector(sensorManager, acceleration ->
{
            DevicePolicyManager devicePolicyManager = (DevicePolicyManager)
getSystemService(Context.DEVICE_POLICY_SERVICE);
            devicePolicyManager.lockNow();
        });
        abruptMotionDetector.register();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    private Notification createNotification() {
        String channelId = "";
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            channelId = createNotificationChannel("tfm.security_service",
"SecurityService");
```

```
        }

        NotificationCompat.Builder builder = new NotificationCompat.Builder(this,
channelId);
        builder.setCategory(Notification.CATEGORY_SERVICE);

        return builder.build();
    }

    @TargetApi(Build.VERSION_CODES.O)
    @RequiresApi(api = Build.VERSION_CODES.N)
    private String createNotificationChannel(String channelId, String channelName) {
        NotificationChannel channel = new NotificationChannel(channelId, channelName,
NotificationManager.IMPORTANCE_HIGH);
        NotificationManager notificationManager = (NotificationManager)
getApplicationContext().getSystemService(Context.NOTIFICATION_SERVICE);
        channel.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);
        notificationManager.createNotificationChannel(channel);
        return channelId;
    }
}
```

6 Conclusiones

En los dispositivos móviles existen diferentes herramientas para poder proteger un dispositivo móvil frente a un robo, además, cada día aparecen nuevas herramientas que pueden ser utilizadas como técnicas de detección y de defensa.

Sin embargo, en los dispositivos móviles se suele llega tarde. Un claro caso es el uso de reconocimiento facial, hace tiempo que es una técnica bastante accesible y no es hasta hace poco tiempo que está empezando a incorporarse en algunos dispositivos.

Además, en el caso de Android, hasta hace poco tiempo no se protegían los datos del usuario de manera predeterminada, de manera que bastaba con conectar el dispositivo a un ordenador para tener acceso completo a toda la información.

Y aunque ahora la seguridad en los dispositivos móviles está empezando a tener más importancia, existen diferentes posibilidades, mediante el uso de los sensores y otras herramientas disponibles, que no han sido exploradas.

De los objetivos planteados en este trabajo se han logrado completar con éxito los siguientes:

- Realizar un estudio de los diferentes sensores disponibles en un dispositivo móvil.

- Reconocer los diferente datos que se pueden obtener de un dispositivo móvil para detectar intrusos.

- Obtener información sobre la API de Android y los posibles usos a aplicar en un sistema de detección de intrusos.

- Ser capaz de identificar a un intruso con un teléfono móvil Android mediante en el uso de los sensores.

- Implementar diferentes mecanismos de seguridad una vez se ha detectado un intruso.

En base a estos objetivos y gracias al estudio realizado, se ha implementado una aplicación que, haciendo uso de la API de Android, es capaz de detectar intrusos con la cámara del dispositivo. Se han implementado tres técnicas de defensa diferentes: notificaciones, bloqueo automático y alerta sonora.

Sin embargo, existe un objetivo que no se ha podido realizar:

- Entrenar y aplicar diferentes algoritmos de aprendizaje automático.

Durante el estudio realizado se han realizado pruebas para intentar perfilar a un usuario basándose en los sensores de movimiento, sin embargo, este objetivo se ha descartado debido a su complejidad y tratarse de un trabajo centrado en la seguridad.

7 Bibliografía

[1] RAE (2019). «Diccionario de la lengua española» - Edición del Tricentenario. [On-line] Dle.rae.es. Disponible en: https://dle.rae.es/srv/search?m=30&w=privacidad [Accedido el 19 Jun. 2019].

[2] StatCounter (2019). Mobile Operating System Market Share Worldwide | StatCounter Global Stats. [On-line] StatCounter Global Stats. Disponible en: http://gs.statcounter.com/os-market-share/mobile/worldwide.

[3] Android Open Source Project (2019). Encryption | Android Open Source Project. [On-line] Android Open Source Project. Disponible en: https://source.android.com/security/encryption [Accedido el 23 Jul. 2019].

[4] Wipo.int. (2018). US20190050058 MOBILE COMMUNICATIONS DEVICE WITH ADAPTIVE FRICTION OF THE HOUSING. [On-line] Disponible en: https://patentscope.wipo.int/search/en/detail.jsf?docId=US237391812 [Accedido el 23 Jul. 2019].

[5] Shumailov, I., Simon, L., Yan, J. and Anderson, R. (2019). Hearing your touch: A new acoustic side channel on smartphones. [On-line] arXiv.org. Disponible en: https://arxiv.org/abs/1903.11137?mod=article_inline [Accedido el 31 Jul. 2019].

[6] Google (2019). Sensors | Android Developers. [On-line] Android Developers. Disponible en: https://developer.android.com/guide/topics/sensors [Accedido el 25 Jul. 2019].

[7] Google (2014). 3.2: Motion and position sensors · GitBook. [On-line] Github.io. Disponible en: https://google-developer-training.github.io/android-developer-advanced-course-concepts/unit-1-expand-the-user-experience/lesson-3-sensors/3-2-c-motion-and-position-sensors/3-2-c-motion-and-position-sensors.html [Accedido el 25 Jul. 2019].

[8] Rotoview.com. (2015). Gravity Sensor in Smartphones and Tablets. [On-line] Disponible en: https://rotoview.com/gravity_sensor.htm [Accedido el 25 Jul. 2019].

[9] Google (2019). Location and context overview | Android Developers. [On-line] Android Developers. Disponible en: https://developer.android.com/training/location [Accedido el 29 Jul. 2019].

[10] Google Developers. (2019). Activity Recognition API | Google Developers. [On-line] Disponible en: https://developers.google.com/location-context/activity-recognition/ [Accedido el 30 Jul. 2019].

[11] Dalal, N. and Triggs, B. (n.d.). Histograms of Oriented Gradients for Human Detection. [On-line] Disponible en: http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf. [Accedido el 1 Ago. 2019]

[12] Geitgey, A. (2016). Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning. [On-line] Medium. Disponible en: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78 [Accedido el 1 Ago. 2019]

[13] Kazemi, V. and Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. 2014 IEEE Conference on Computer Vision and Pattern Recognition. [On-line] Disponible en: http://www.csc.kth.se/~vahidk/papers/KazemiCVPR14.pdf. [Accedido el 3 Ago. 2019]

[14] Firebase. (2019). ML Kit for Firebase | Firebase. [On-line] Disponible en: https://firebase.google.com/docs/ml-kit [Accedido el 6 Ago. 2019].

[15] TensorFlow. (2019). TensorFlow Lite | TensorFlow Lite | TensorFlow. [On-line] Disponible en: https://www.tensorflow.org/lite [Accedido el 6 Ago. 2019].

[16] Android Developers. (2019). Device administration overview | Android Developers. [On-line] Disponible en: https://developer.android.com/guide/topics/admin/device-admin.html [Accedido el 7 Ago. 2019].